

AO-A101 710

INTEGRATED INFORMATION SUPPORT SYSTEM (IIS) VOLUME 3
COMMON DATA MODEL 5. (U) GENERAL ELECTRIC CO
SCHENECTADY NY PRODUCTION RESOURCES CONSU..

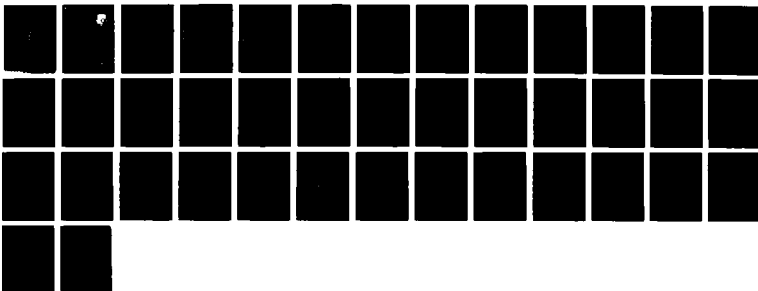
1/1

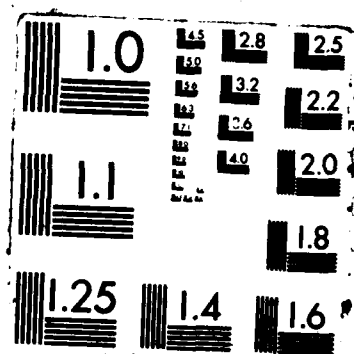
UNCLASSIFIED

H LOOHIS ET AL. 01 NOV 85 DS-620141310

F/G 12/5

NL





DTIC FILE COPY

**AFVAL-TR-86-4006
Volume V
Part 26**

AD-A181 710



**INTEGRATED INFORMATION
SUPPORT SYSTEM (IISS)
Volume V - Common Data Model Subsystem
Part 26 - Distributed Request Supervisor
Development Specification**

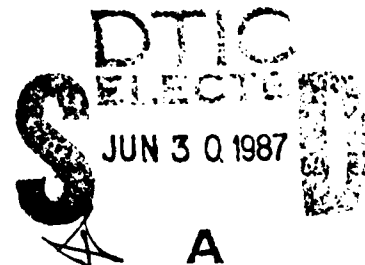
**General Electric Company
Production Resources Consulting
One River Road
Schenectady, New York 12345**

**Final Report for Period 22 September 1980 - 31 July 1985
November 1985**

Approved for public release; distribution is unlimited.

PREPARED FOR:

**MATERIALS LABORATORY
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AFB, OH 45433-6533**

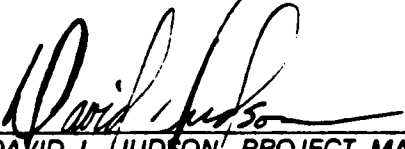


NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed by the Office of Public Affairs (ASD/PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.




DAVID L. JUDSON, PROJECT MANAGER
AFWAL/MLTC
WRIGHT PATTERSON AFB OH 45433

5 Aug 1986

DATE

FOR THE COMMANDER:



GERALD C. SHUMAKER, BRANCH CHIEF
AFWAL/MLTC
WRIGHT PATTERSON AFB OH 45433

7 Aug 86

DATE

"If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify AFWAL/MLTC, W-PAFB, OH 45433 to help us maintain a current mailing list."

Copies of this report should not be returned unless return is required by security considerations contractual obligations, or notice on a specific document.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

1 November 1985

REPORT DOCUMENTATION PAGE

A181 710

1a REPORT SECURITY CLASSIFICATION Unclassified		1b RESTRICTIVE MARKINGS	
2a SECURITY CLASSIFICATION AUTHORITY		3 DISTRIBUTION AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.	
2b DECLASSIFICATION/DOWNGRADING SCHEDULE			
4. PERFORMING ORGANIZATION REPORT NUMBER(S)		5. MONITORING ORGANIZATION REPORT NUMBER(S) AFVAL-TR-86-4006 Vol V, Part 26	
6a NAME OF PERFORMING ORGANIZATION General Electric Company Production Resources Consulting	6b OFFICE SYMBOL (If applicable) AFVAL/MLTC	7a NAME OF MONITORING ORGANIZATION AFVAL/MLTC	
6c ADDRESS (City, State and ZIP Code) 1 River Road Schenectady, NY 12345		7b ADDRESS (City, State and ZIP Code) VPAFB, OH 45433-6533	
8a NAME OF FUNDING/SPONSORING ORGANIZATION Materials Laboratory Air Force Systems Command, USAF	8b OFFICE SYMBOL (If applicable) AFVAL/MLTC	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F33615-80-C-5185	
8c ADDRESS (City, State and ZIP Code) Wright-Patterson AFB, Ohio 45433		10 SOURCE OF FUNDING NOS.	
		PROGRAM ELEMENT NO. 78011F	PROJECT NO. 7500
		TASK NO. 62	WORK UNIT NO. 01
11. TITLE (Include Security Classification) (See Reverse)			
12. PERSONAL AUTHOR(S) Loomis, Mary and Lipp, Mark			
13a TYPE OF REPORT Final Technical Report	13b TIME COVERED 22 Sept 1980 - 31 July 1985	14. DATE OF REPORT (Yr., Mo., Day) 1985 November	15. PAGE COUNT 45
16. SUPPLEMENTARY NOTATION ICAM Project Priority 6201		The computer software contained herein are theoretical and/or references that in no way reflect Air Force-owned or -developed computer software.	
17. ESDAT: CODES		18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)	
FIELD	GROUP	SUB GR	
1908	0905		
19. ABSTRACT (Continue on reverse if necessary and identify by block number) The Common Data Model Processor (CDMP) is a mechanism by which application programs can retrieve and update data without knowing where or how the data are stored. An application program poses requests to the CDMP, which processes those requests against the databases in which the relevant data are stored and then returns the results to the application program. The Neutral Data Manipulation Language (NDML) is the means for posing requests to the CDMP. One component of the CDMP, called the NDML Precompiler, generates various programs (request processors or RPs, RP drivers, CS-ES transformers, and local subroutine callers) that are tailored to satisfy the NDML requests in a particular application program. Another component of the CDMP, called the Distributed Request Supervisor (DRS), coordinates the operation of these generated programs when they are invoked by the application program at run-time. This development specification describes the functions, performance, environment, interfaces, and design requirements of the Distributed Request Supervisor.			
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED <input checked="" type="checkbox"/> SAME AS RPT. <input type="checkbox"/> DTIC USERS <input type="checkbox"/>		21. ABSTRACT SECURITY CLASSIFICATION Unclassified	
22a NAME OF RESPONSIBLE INDIVIDUAL David L. Judson		22b TELEPHONE NUMBER (Include Area Code) 813-255-8976	22c OFFICE SYMBOL AFVAL/MLTC

DD FORM 1473, 83 APR

EDITION OF 1 JAN 73 IS OBSOLETE.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

11. Title

Integrated Information Support System (IISS)
Vol V - Common Data Model Subsystem
Part 26 - Distributed Request Supervisor
Development Specification



Accession For	
NTIC GRA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A1	

PREFACE

This development specification covers the work performed under Air Force Contract F33615-80-C-5155 (ICAM Project 6201). This contract is sponsored by the Materials Laboratory, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio. It was administered under the technical direction of Mr. Gerald C. Shumaker, ICAM Program Manager, Manufacturing Technology Division, through Project Manager, Mr. David Judson. The Prime Contractor was Production Resources Consulting of the General Electric Company, Schenectady, New York, under the direction of Mr. Alan Rubenstein. The General Electric Project Manager was Mr. Myron Hurlbut of Industrial Automation Systems Department, Albany, New York.

Certain work aimed at improving Test Bed Technology has been performed by other contracts with Project 6201 performing integrating functions. This work consisted of enhancements to Test Bed software and establishment and operation of Test Bed hardware and communications for developers and other users. Documentation relating to the Test Bed from all of these contractors and projects have been integrated under Project 6201 for publication and treatment as an integrated set of documents. The particular contributors to each document are noted on the Report Documentation Page (DD1473). A listing and description of the entire project documentation system and how they are related is contained in document FTR620100001, Project Overview.

The subcontractors and their contributing activities were as follows:

TASK 4.2

Subcontractors

Role

Boeing Military Aircraft
Company (BMAC)

Reviewer

D. Appleton Company
(DAGOM)

Responsible for IDEF support,
state-of-the-art literature
search

General Dynamics/
Ft. Worth

Responsible for factory view
function and information
models

DS 620141310
1 November 1985

Subcontractors

Role

Illinois Institute of
Technology

Responsible for factory view
function research (IITRI)
and information models of
small and medium-size business

North American Rockwell

Reviewer

Northrop Corporation

Responsible for factory view
function and information
models

Pritsker and Associates

Responsible for IDEF2 support

SofTech

Responsible for IDEF0 support

TASKS 4.3 - 4.9 (TEST BED)

Subcontractors

Role

Boeing Military Aircraft
Company (BMAC)

Responsible for consultation on
applications of the technology
and on IBM computer technology.

Computer Technology
Associates (CTA)

Assisted in the areas of
communications systems, system
design and integration
methodology, and design of the
Network Transaction Manager.

Control Data Corporation
(CDC)

Responsible for the Common Data
Model (CDM) implementation and
part of the CDM design (shared
with DACOM).

D. Appleton Company
(DACOM)

Responsible for the overall CDM
Subsystem design integration and
test plan, as well as part of
the design of the CDM (shared
with CDC). DACOM also
developed the Integration
Methodology and did the schema
mappings for the Application
Subsystems.

DS 620141310
1 November 1985

Subcontractors

Role

Digital Equipment
Corporation (DEC)

Consulting and support of the
performance testing and on DEC
software and computer systems
operation.

McDonnell Douglas
Automation Company
(McAuto)

Responsible for the support and
enhancements to the Network
Transaction Manager Subsystem
during 1984/1985 period.

On-Line Software
International (OSI)

Responsible for programming the
Communications Subsystem on the
IBM and for consulting on the
IBM.

Rath and Strong Systems
Products (RSSP) (In 1985
became McCormack & Dodge)

Responsible for assistance in
the implementation and use of
the MRP II package (PIOS) that
they supplied.

SofTech, Inc.

Responsible for the design and
implementation of the Network
Transaction Manager (NTM) in
1981/1984 period.

Software Performance
Engineering (SPE)

Responsible for directing the
work on performance evaluation
and analysis.

Structural Dynamics
Research Corporation
(SDRC)

Responsible for the User
Interface and Virtual Terminal
Interface Subsystems.

Other prime contractors under other projects who have
contributed to Test Bed Technology, their contributing
activities and responsible projects are as follows:

<u>Contractors</u>	<u>ICAM Project</u>	<u>Contributing Activities</u>
Boeing Military Aircraft Company (BMAC)	1701, 2201, 2202	Enhancements for IBM node use. Technology Transfer to Integrated Sheet Metal Center (ISMC)

DS 620141310
1 November 1985

<u>Contractors</u>	<u>ICAM Project</u>	<u>Contributing Activities</u>
Control Data Corporation (CDC)	1502, 1701	IISS enhancements to Common Data Model Processor (CDMP)
D. Appleton Company (DACOM)	1502	IISS enhancements to Integration Methodology
General Electric	1502	Operation of the Test Bed and communications equipment.
Hughes Aircraft Company (HAC)	1701	Test Bed enhancements
Structural Dynamics Research Corporation (SDRC)	1502, 1701, 1703	IISS enhancements to User Interface/Virtual Terminal Interface (UI/VTI)
Systran	1502	Test Bed enhancements. Operation of Test Bed.

TABLE OF CONTENTS

		<u>Page</u>
SECTION 1.0	SCOPE	1-1
1.1	Identification	1-1
1.2	Functional Summary	1-4
SECTION 2.0	DOCUMENTS	2-1
2.1	Applicable Documents	2-1
2.2	Terms and Abbreviations	2-2
SECTION 3.0	REQUIREMENTS	3-1
3.1	Computer Program Definition	3-1
3.1.1	System Capacities	3-1
3.1.2	Interface Requirements	3-1
3.1.3	Design/Implementation Differences .	3-2
3.2	Detailed Functional Requirements	3-3
3.2.1	Function DRS1 Initiate Subtransaction Processing	3-3
3.2.2	Function DRS2 Schedule Stages	3-7
3.2.3	Function DRS3 Initiate CS/ES Transform Processing	3-18
3.3	Special Requirements	3-20
3.4	Human Performance	3-20
3.5	Database Requirements	3-20
3.6	Adaptation Requirements	3-20
SECTION 4.0	QUALITY ASSURANCE PROVISION	4-1
SECTION 5.0	PREPARATION FOR DELIVERY	5-1

LIST OF FIGURES

<u>Figure</u>	<u>Title</u>	<u>Page</u>
1-1	AO of the CDMP Configuration Items	1-2
1-2	DRS Module Interaction	1-6

SECTION 1

SCOPE

1.1 Identification

This specification establishes the performance, development, test, and qualification requirements of a collection of computer programs identified as Configuration Item "Distributed Request Supervisor."

This CI constitutes one of the major subsystems of the "Common Data Model Processor" (CDMP) which is described in the System Design Specification (SDS) for the ICAM Integrated Support System (IISS). The CDMP scope is based on a logical concept of subsystem modules that interface with other external systems of the IISS. The CDMP has been decomposed into three configuration items: the Precompiler, the Distributed Request Supervisor, and the Aggregator. The scope of the CDMP and its configuration items are described in Figure 1-1 and the following narrative.

Common Data Model Processor (CDMP)

The CDMP consists of three CIs that manage users' accesses to distributed databases in IISS. Input to the CDMP consists of user transactions, which may be in the form of neutral data manipulation language (NDML) commands embedded in COBOL host programs or NDML commands phrased as stand-alone requests. These development specifications address only the management of embedded NDML commands.

The Precompiler CI parses the application program source code, identifying NDML commands. It applies external-schema-to-conceptual-schema transforms on the NDML command, and decomposes the conceptual schema command into single database requests. These single database requests are each transformed into programs (called Request Processors) to access the specific databases to retrieve or update the data as required by the NDML command. The NDML commands in the application source program are replaced by function calls which, when executed, will activate the run-time query evaluation processes associated with the particular NDML command.

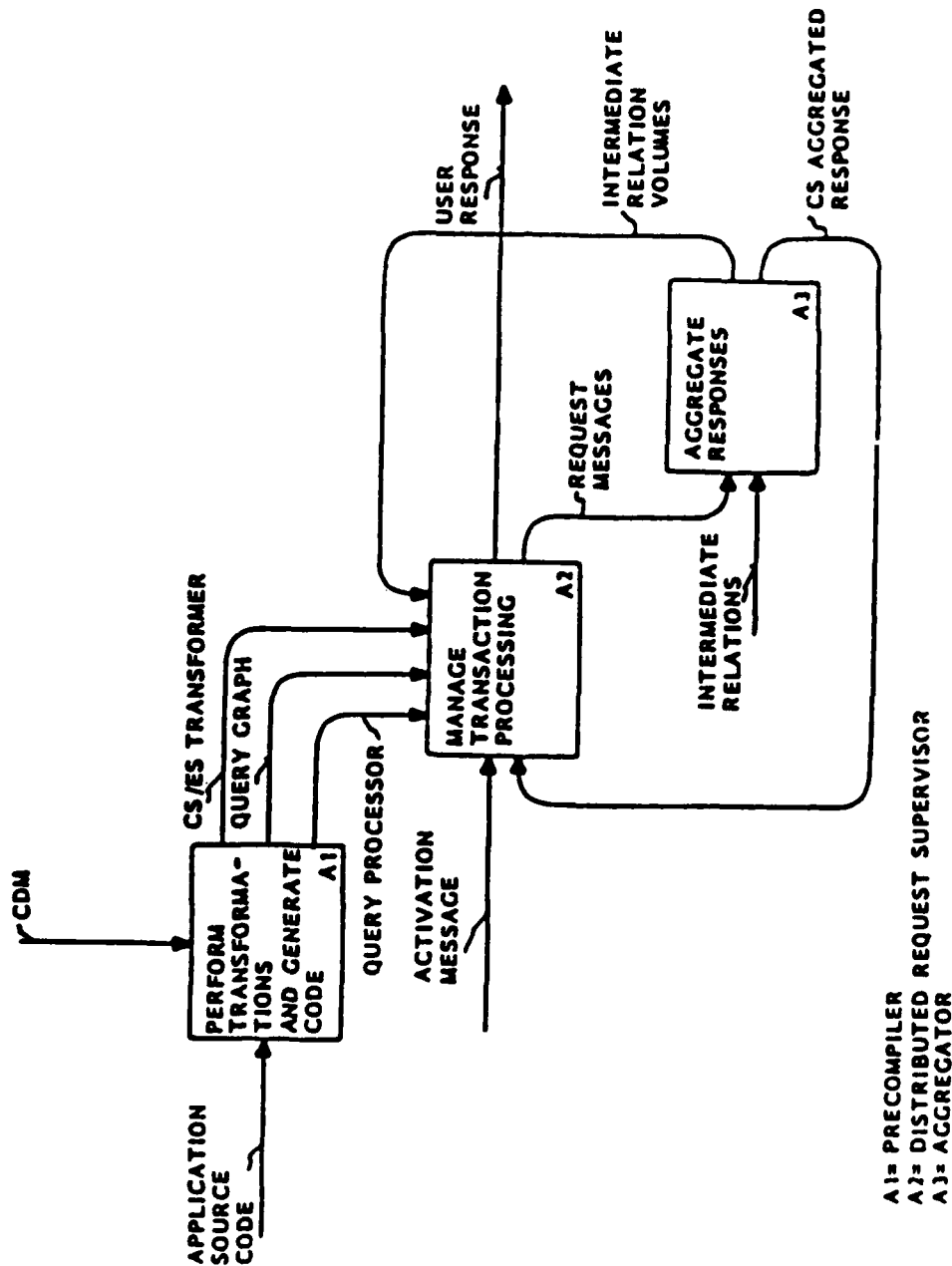


Figure 1-1. A0 of the CDNP Configuration Items

The Precompiler also generates a CS/ES Transformer program which will take the final result of the query, stored in a file as a conceptual schema relation, and transform it into the appropriate external schema relation.

Finally, the Precompiler generates a Join Query Graph and Result Field Table, which are used by the Distributed Request Supervisor during the run-time evaluation of the query.

The Distributed Request Supervisor (DRS) CI is responsible for coordination of the run-time activity associated with the evaluation of an NDML command. It is activated by the application program, which sends it the names and locations of the Request Processors to be activated, along with run-time parameters which are to be sent to the Request Processors. The DRS activates the Request Processors, sending them the run-time parameters. The results of the Request Processor executions are stored as files, in the form of conceptual schema relations, on the hosts which executed the Request Processors. Using the Join Query Graph, transmission cost information, and data about intermediate results, the DRS determines a good strategy for combining the intermediate results of the NDML command. It issues the appropriate file transfer requests, activates Aggregators to perform join, union, and not-in-set operations, and activates the appropriate CS/ES Transformer program to transform the final results. Finally, the DRS notifies the application program that the query is completed, and sends it the name of the file which contains the results of the query.

The Aggregator CI is activated by the DRS. An instance of the Aggregator is executed for each join, union, or not-in-set performed. It is passed information describing the union or join to be performed, including the file names containing the operands of the union or join. The DRS ensures that these files already exist on the host that is executing the particular Aggregator program. The Aggregator performs the requested union or join, storing the results in a file, whose name was specified by the DRS, and is located on the host executing the Aggregator.

The CDMP provides the application programmer with important capabilities to:

1. Request database accesses in a non-procedural data manipulation language (the NDML) that is independent of the DML of any particular Data Base Management

System (DBMS).

2. Request database access using a DML that specifies accesses to a set of related records rather than to individual records, i.e., using a relational DML.
3. Request access to data that are distributed across multiple databases with a single DML command, without knowledge of data locations or distribution details.

Information about external schemas, the conceptual schema, and internal schemas (including data locations) are provided by CDM access to the Common Data Model (CDM) database. The CDM is a relational database of metadata pertaining to IISS. It is described by the CDM1 information model using IDEF1.

1.2 Functional Summary

The overall objectives of this CI are to:

- a. Determine the appropriate sequence of inter-database JOIN, UNION and NOT-IN-SET operations required to produce the result for a multi-database transaction.
- b. Coordinate and control the interactions among a user's Application Process (AP), the generated Request Processors (RP) and the Aggregator(s) for both single- and multi-database transactions.

Determination of JOIN, UNION, NOT-IN-SET

The DRS will calculate costs for each inter-site join, union, and not-in-set possibility, select the alternative with minimum cost and will generate the appropriate sequence of join, union, and not-in-set operations that will collapse the intermediate relations into the proper destination relation. The sequence is generated at run-time by the DRS at the node of the transaction's originating AP cluster.

The Distributed Request Supervisor receives information about a set of intermediate relations, which are the result of processing portions of a Transaction at the local databases. Once all local processing is complete, the intermediate relations must be joined together. The Distributed Request Supervisor solves the problem of determining in which order these relations should be combined, which includes the

sequencing of transmission of relations from one database to another in order to perform the operations, in such a way as to minimize the amount of data transferred. The operations are performed by the Aggregator CI.

In order to process a transaction efficiently, the Distributed Request Supervisor determines all possible inter-database operations and calculates the transmission costs for each possibility. It selects the operation with the least cost, sends the appropriate transmission commands to the identified Aggregator site, and updates information tables as each is performed. If a join is chosen as the next step to schedule, the two relations may not be partitions; only "whole" relations are joined. After the last operation is performed, the resulting relation is transmitted to the site at which the result is to appear. A CS/ES Transform process is then initialized to perform the required transformations.

AP/RP/Aggregator Coordination

Each user AP that contains NDML requests has a copy of the DRS, which it calls as a subroutine. All these copies are exactly the same. Each copy is responsible for the coordination and control of all the local and remote RPs, Aggregators, and CS-ES Transformers that are used to process the NDML requests from its user AP (see Figure 2-1). A local RP is one that is called as a subroutine by the DRS and that accesses a database on the same node as the user AP. A remote RP is one that is called via the NTM; the database it accesses may be on the same node as the user AP or on a different node. The DRS uses the NTM message handling facility to communicate with any remote RPs and Aggregators. It calls any local RPs and Aggregators and all the CS-ES transformers as subroutines via the Subroutine Caller generated by PRE15. The only function of the Subroutine Caller is to call a subroutine that is designated by the DRS. This allows each copy of the DRS to (indirectly) call a variety of RPs, Aggregators, and CS-ES Transformers while still being identical to all other DRS copies (PRE15 assigns the same name to all the Subroutine Callers). If the DRS called these subroutines directly, the subroutine call statements in each copy would have to be different from those in any other copy.

The initiation of a transaction occurs when the AP sends a message to the DRS to initiate activity on the specified transaction. The DRS will then initiate the proper DRS tables and Request Processors (RPs) in preparation for the first NDML

request from the AP. When the AP makes an NDML request, the AP will 'go to sleep' and wait for the DRS response. The DRS will activate/reactivate the proper RP(s). The DRS will wait until the (RPs) have completed, then will decide if an Aggregator needs to be called. If so, the DRS will do so and wait until it receives a message from the Aggregator indicating completion. The DRS will then either return directly to the AP or call the CS/ES Transformer, whichever is appropriate.

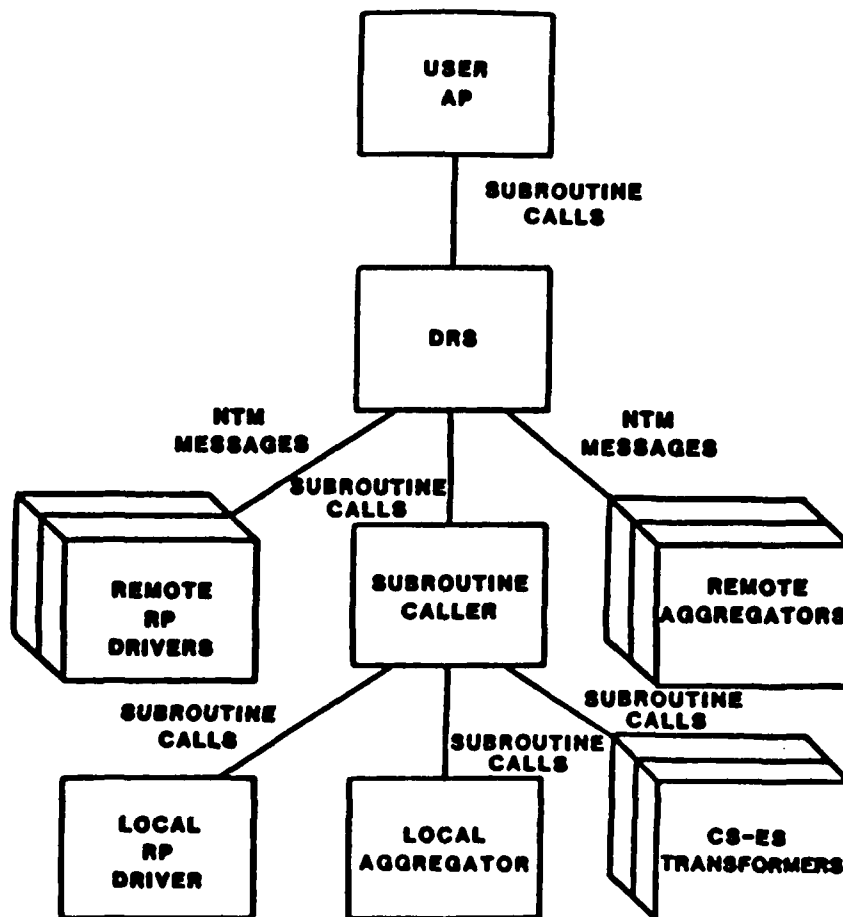


Figure 1-2. DRS Module Interaction

Since the proper execution of recovery units in the local DBMSs requires that a specific AP communicate with a consistent instantiation of a RP during the life of the AP recovery unit, the DRS must guarantee that, once an AP initiates a recovery unit, the RP to which the first local DML request was sent (within the confines of the recovery unit) must be the RP to which all further local DML requests (for the specific DBMS/node) are sent, until the recovery unit termination request is received from the AP. Therefore, the DRS will maintain a table of the RPs for each AP with which the DRS is communicating, regardless of the type of NDML verbs being executed by the AP.

Compile-time Activities

Compile-time activities include building skeleton process information tables. At run-time, the Distributed Request Supervisor sends messages for the local Request Processors to begin processing. As the local Request Processors finish their subtransactions, they send back information for the Distributed Request Supervisor to use in deciding the sequence of steps to be taken. The Distributed Request Supervisor then, as needed, initiates file transfer requests, activates appropriate Aggregators, and eventually activates the appropriate CS/ES Transformer to transform the final result relation if a SELECT was requested.

The major functions to be described in this document for this CI are:

- DRS1: Initiate Subtransaction Processing
- DRS2: Schedule Stages
- DRS3: Initiate CS/ES Transform Processing

SECTION 2

DOCUMENTS

2.1 Applicable Documents

Following is a list of applicable documents relating to this Computer Program Development Specification for the system identified as the Common Data Model Processor (CDMP) Distributed Request Supervisor.

Related ICAM Documents included:

UM620141001	<u>CDM Administrator's Manual</u>
TBM620141000	<u>CDM1, An IDEF1 Model of the Common Data Model</u>
UM620141100	<u>Neutral Data Definition Language (NDDL) User's Guide</u>
PRM620141200	<u>Embedded NDML Programmer's Reference Manual</u>
UM620141002	<u>ICAM Definition Method for Data Modeling (IDEF1 - Extended)</u>
DS620141200	<u>Development Specification for the IISS NDML Precompiler Configuration Item</u>
DS620141320	<u>Development Specification for the IISS Aggregator Configuration Item</u>

Other references include:

Astrahan, M.M. et al., "System R: Relational Approach to Database Management," ACM Transactions on Database Systems, Vol. 1, No. 2, June 1976, pp. 97-137.

Bernstein, P.A. and Chiu, D.M., "Using Semi-Joins to Solve Relational Queries," Journal for the Association for Computing Machinery, Vol. 28, No. 1, January 1981, pp. 25-40.

Bernstein, P.A. et al., "Query Processing in a System for Distributed Databases (SDD-1)," ACM Transactions on Database Systems, Vol. 6, No. 4, December 1981, pp. 602-625.

Chang, J.M. "A Heuristic Approach to Distributed Query Processing," Proceedings of the Eighth International Conference on Very Large Data Bases, Mexico City, September 1982, pp. 54-61.

Epstein, R., M. Stonebraker, and E. Wong. "Distributed Query Processing in a Relational Database System," Proceedings of the ACM SIGMOD International Conference, Austin, June 1978, pp. 169-180.

Hevner, A.R. and S.B. Yao. "Query Processing in Distributed Database Systems," IEEE Transactions on Software Engineering, May 1979, pp. 177-187.

Rothnie, J.B. et al. "Introduction to a System for Distributed Databases," ACM Transactions on Database Systems, Vol. 5, No. 1, March 1980, pp. 1-17.

Rothnie, J.B. and N. Goodman. "A Survey of Research and Development in Distributed Database Management," Proceedings Third International Conference on Very Large Databases, Tokyo, 1977, pp. 48-62.

Takizawa. "Distributed Database System - JDDBS-1," JIPDEC, Japan, 1982.

Wong, E. and K. Youssefi. "Decomposition - A Strategy for Query Processing," ACM Transactions on Database Systems, Vol. 1, No. 3, September, 1976, pp. 223-241.

2.2 Terms and Abbreviations

The following acronyms are used in this document:

APL Attribute Pair List

AUC Attribute Use Class

CDMP Common Data Model Processor

CI Configuration Item

DS 620141310
1 November 1985

CS Conceptual Schema

DML Data Manipulation Language

DRS Distributed Request Supervisor
 (previously SS: Stager/Scheduler)

ES External Schema

ICAM Integrated Computer Aided Manufacturing

IS Internal Schema

NDML Neutral Data Manipulation Language

RFT Result Field Table

RP Request Processor
 (previously QP: Query Processor)

SDS System Design Specification

SECTION 3

REQUIREMENTS

3.1 Computer Program Definition

3.1.1 System Capacities

The DRS must operate within the capacity of the host computer and is functionally dependent upon NTM Services.

3.1.2 Interface Requirements

3.1.2.1 Interface Blocks

This CI is the mechanism that determines the order of aggregating intermediate results of accesses to distributed databases, and coordinates interactions among the AP/RP/DRS/Aggregators during updates.

There is a Distributed Request Supervisor program for each site or host in the IISS network. An instance of the Distributed Request Supervisor (DRS) program runs as a subroutine to each user AP that contains NDML commands. This DRS takes on the role of master control program for all the transactions from that user AP. Instances of Distributed Request Supervisor at other sites become the master control programs for transactions initiated at those sites.

The Distributed Request Supervisor CI has responsibility for run-time scheduling of activities comprising distributed database accesses and updates. It initiates local Request Processors and receives replies when they have completed. It sends subtransactions to appropriate application clusters, initiates file transfer requests to transmit intermediate relations, and initiates corresponding Aggregators to perform join, union, and not-in-set operations on intermediate relations. Finally, in an access operation, it initiates the appropriate CS/ES Transformer process to transform the final results.

The interfaces of each Distributed Request Supervisor include input in the form of messages indicating completion of activities under the Distributed Request Supervisor control and the Join Query Graph from the Precompiler CI. Outputs are in the form of "staging sequences" that direct activities of other

run-time modules.

3.1.2.2 Detail Interface Definition

The specific interface relationships of this CI to other CIs and modules are described in detail for appropriate functions in Section 3.2.

The DRS depends heavily upon three capabilities of the NTM, and, without these facilities, will not properly function.

- a. If a process dies or is killed, the NTM must notify, via an unsolicited message, the parent process.
- b. If a process dies or is killed, the NTM must kill all processes which are children of the dead process. The children processes must not be given the option of continuing or dying; they must be killed.
- c. The DRS will communicate with other processes via messages which are guaranteed to be delivered. The NTM must provide this facility in an efficient manner, and must include a mechanism to properly handle the node-dropping/node-returning/node-isolated problems.

3.1.3 Design/Implementation Differences

This section describes the significant differences between the design of the Distributed Request Supervisor (DRS) that is documented in this Development Specification and the software that has been produced to implement the DRS. This section is not concerned with minor differences, such as the exact structure of tables that are passed from one module to another within the DRS.

The only difference is the way in which CS-ES transformers are invoked by user APs at run-time. The design indicates that they are invoked via the Distributed Request Supervisor (DRS), but the Precompiler software generates code into the user APs to invoke them directly. This is possible because every CS-ES transformer must run on the same host computer as its user AP. Consequently, the DRS (and the NTM) are not needed. This was not foreseen when the design was prepared and time did not permit the design to be changed later. This difference affects the PRE10 and PRE15 modules of the Precompiler as well as the DRS. The design of PRE10 should indicate that code is

generated into the user AP source program to invoke each CS-ES transformer directly as a subroutine, rather than via the DRS and a local subroutine caller (LSC). The design of PRE15 should not indicate that LSCs are generated containing code to invoke CS-ES transformers. The design of the DRS should not indicate that DRSS are involved in invoking CS-ES transformers.

3.2 Detailed Functional Requirements

The following subsections respectively document each of the Distributed Request Supervisor major function identified in Section 1.2.

3.2.1 Function DRS1: Initiate/Resume Subtransaction Processing

This function directs appropriate Request Processors to begin or resume processing. The Request Processors were built by the Precompiler CI and are co-located with their target databases.

3.2.1.1 Inputs

Inputs to this function are:

- The program ID (PID) and runtime parameters for each Request Processor which is to be activated for the NDML request. Four data items are input for each Request Processor to be activated: The PID, a code indicating whether to use the LSC or the NTM, a string containing the corresponding runtime parameters, and the length of the string.

Run-time parameters are to be applied by the Request Processors to the subtransactions. These parameters are the values of the COBOL variables that were part of

the NDML query. The first variable must contain the CASE statement number generated by the Precompiler.

- Responses from the NTM as a result of the DRS START-LOCAL requests for the RPs. These responses will contain the Logical Channel ID and the local process ID for the initiated RPs.
- Join Query Graph corresponding to the NDML request being processed.

- An attribute pair list. This list contains information concerning the join fields of the join edges of the Join Query Graph. This list will not be present for an update request.
- A Result Field Table. This table contains information about the result and join attributes of the query. This list will not be present for an update request.
- A Request Processor Information Table (RPIT) (see 3.2.2.4).

The first input is received from the Generate Request Processor function (PRE9) of the Precompiler CI. The other inputs are from the Decompose CS NDML function (PRE5) of the Precompiler CI.

A join query graph (JQG) corresponds to a CS NDML verb. Each node of a JQG represents an intermediate relation that will result from processing a single subtransaction, which accesses one database. Each edge of a JQG represents an inter-database join, union, or not-in-set operation between two relations. The set of edges represents the join, union, and not-in-set operations that in combination will result in the response to the CS NDML transaction. The format of the JQG is a table, with an entry for each edge of the graph. Each entry contains the following information:

rel1 rel2 edge-type attr-ptr PID1 PID2

where:

rel1	-	one of the edge nodes
rel2	-	the other node
edge-type	-	(join) = 4
		(union) = 5
		(not-in-set) = 6
attr-ptr	-	A pointer into the attribute pair list. It is null if the edge type is UNION.
PID1	-	The PID of the Request processor which will create rel1.
PID2	-	The PID of the Request Processor which will create rel2.

The format of the attribute pair list (APL) is a group of linked lists of attribute pairs, one linked list for each edge of the JQG. Each entry in each list contains the following:

rel1 rel2 attr1 attr2 link

where:

rel1	-	the name of one edge node
rel2	-	the name of the other edge node
attr1	-	the Attribute Use Class number (AUC) of the attribute of rel1 which is participating in the join with attr2.
attr2	-	The AUC of the attribute of rel2 which is participating in the join with attr1.
link	-	a pointer to the next entry in the list. (A join can have more than one join field pair). The field is null if there are no more entries in the list.

Each entry in the Result Field Table (RFT) has the following format:

rel attr type size nd PID is-ptr

where:

rel1	-	the name of the relation (subtransaction) that contains the field
attr	-	the Attribute Use Class number (AUC) of the field
type	-	the type of the field (alphabetic, numeric, etc.)
size	-	the size, in bytes, of the field
nd	-	the number of decimal places maintained in the field
PID	-	not used
is-ptr	-	not used

3.2.1.2 Processing

This function starts the processing of the subtransactions that comprise a distributed database access or update. The following steps are performed by this function.

1. Initialize the Relation Information Table (RIT). The format of this table is described in Section 3.2.2.4. An entry is placed in the RIT for each relation to be constructed or accessed by a Request Processor. A unique name is generated for each relation and placed

in the RIT. This name must be a legal file name for the file system at the host where the relation will be constructed. In addition, this step must guarantee that the name will be unique over all active queries in the system, including simultaneous instances of the same query.

2. Replace the Rel1 and Rel2 values in the JQG with an index number into the RIT, which corresponds to the entry for the appropriate relation.
3. Replace the Rel values in the RFT with the corresponding index number into the RIT, if this request is an access request.
4. If the RPIT (Request Processor Information Table) (see 3.2.2.4) has not been initialized for the specific instance of the AP, or if an existing RPIT has been set to "uninitialized" by the appearance of an NDML recovery unit termination request, the RPIT is now initialized by establishing such a table with one entry per RP name contained in the input message stream. If the RPIT had already been established, go on.
5. If Step 4 initialized the RPIT:

First, initiate each RP with RP-call-type "R" in the QIT by issuing a STARTLOCAL message to the NTM. Then, initiate each with RP-call-type "L" by calling it as a subroutine. The information given to the subroutine or the NTM will include the RP PID and the runtime parameters for each RP. The CASE statement number passed to the RP during this initiation phase is to be zero.

The NTM will return, for each remote RP initiated, the Logical Channel ID and the local process ID. The Logical Channel ID is to be placed into the corresponding entry in the RIT.

The local process ID for each remote RP is to be placed into the corresponding entry in the RPIT. The Host ID of the corresponding entries in the RIT are to be set to the Host ID of the hosts upon which each local or remote RP is running.

The status fields in the RIT are to be set to "BUSY".

If Step 4 did not initialize the RPIT:

Update the RPIT with any RPs that have a PID in the input message but do not currently appear in the RPIT. For those RPs added to the QIT, perform the functions stated in the above description for a newly-initiated RPIT.

Processing a subtransaction involves performing local restricts (a.k.a. selects), projects, and single-database joins. These operators have been translated to the DML appropriate for the local DBMS by the NDML/Generic DML Transformer function (PRE7) and the RP Generator function (PRE9) of the Precompiler CI. The local result relations should contain only join attributes and (final) result attributes.

3.2.1.3 Outputs

The outputs of this function are:

- STARTLOCAL messages that activate or resume remote Request Processors. These messages contain the run-time parameters of the subtransaction.
- An initialized RIT, if the request is an access request.
- The modified JQG, if the request is an access request.
- An initialized or modified RPIT.

3.2.2 Function DRS2: Schedule Stages

This function iteratively determines the sequence in which intermediate relations are combined to form the result of a distributed database access. The sequence of join/union/not-in-set activities may include both parallel and serial processing.

3.2.2.1 Inputs

Inputs to this function are:

- Join Request Processor Graph corresponding to the NDML request being scheduled.
- Result Field Table, if the request is an access request.
- CS-ACTION-LIST, if the request is an access request.
- ENDLOCAL messages
- ENDJOIN, ENDUNION, ENDNOTINSET messages
- ENDFILESEND MESSAGES
- Request Processor Information Table

The Join Query Graph (JQG) and Result Field table (RFT) inputs are also inputs to function DRS1 which modified them, and are described in section 3.2.1.1.

The CS-ACTION-LIST (CSAL) is a list of the attributes which will comprise the result relation. The order of this list is the order of the attributes in which the CS/ES Transformer will expect the final results to be. The format of the CSRL is the following:

ent-class auc workptr type size nd

where:

ent-class	=	not used
auc	=	Attribute Use Class of the attribute
workptr	=	not used
type	=	not used
size	=	not used
nd	=	not used

ENDLOCAL messages are issued by the Request Processors and contain information about the intermediate relations that result from local processing. They arrive on the same logical Channel ID as was assigned when the Request Processor was initiated or via a parameter if the RP was called as a sub-routine. ENDLOCAL indicates that processing of a subtransaction has been completed. The form of an ENDLOCAL message is:

ENDLOCAL length

where:

length - the number of tuples in the resultant
relation

ENDJOIN, ENDUNION, and ENDNOTINSET messages are issued by the Aggregator CI and contain information about the relations that result from combining intermediate relations and transmitting them to another application cluster. As in ENDLOCAL messages, they arrive on the same logical channel as was assigned when the Aggregator was invoked or via subroutine parameters.

ENDJOIN indicates that processing of an inter-database join has been completed. The form of an ENDJOIN message is:

ENDJOIN length

where:

length - the number of tuples in the resultant
relation

ENDUNION indicates that the processing of an inter-database union has been completed. The form of an ENDUNION message is:

ENDUNION length

where:

length - the number of tuples in the resultant
relation

ENDNOTINSET indicates that the processing of an inter-database not-in-set has been completed. The form of an ENDNOTINSET message is:

ENDNOTINSET length

where:

length - the number of tuples in the resultant

relation

ENDFILESEND messages are sent by the File Transfer process to acknowledge that a file has been sent as requested.

3.2.2.2 Processing

This function receives information about the results of processing of intermediate results.

The following describes the algorithm used by this module to control the runtime evaluation of a request. The general strategy is to break the request into stages, execute the stages serially, but execute the components of each stage in parallel. The first stage is comprised of all the request processors. Subsequent stages consist of file transfer requests, join requests, union requests, and/or not-in-set requests. Step 4, described below, determines which requests comprise subsequent stages.

Step 1. Initialize Scheduler Tables

Create or update the following Performance Information Tables (PITs):

- a. Read the Transmission Cost Table (TCT) from a file. There is one TCT file at each site. The format of this table is described in Section 3.2.2.4.
- b. For each entry in the RIT, calculate the width of the relation to be constructed by the corresponding request processor. This is calculated by examining the corresponding entries in the RFT. Enter each width into the appropriate RIT entry.
- c. Initialize the Cost Information Table (CIT). Each entry in the CIT corresponds to a candidate Union or Join action. The format of this table is described in Section 3.2.2.4. For each entry in the JQG, there will be two entries in the CIT. The first entry will have rel1 as the source relation and rel2 as the dest relation, and the other entry will have rel2 as the source relation and rel1 as the dest relation. For

each entry placed in the CIT, set the source and dest fields as just described, set the i-p field to null, and set the edge-id field to the appropriate index into the JQG.

Step 2. Process Incoming Messages

This step processes reply messages sent by the Request Processors and Aggregators. The Logical Channel ID, which is a part of each message, is used to locate the entries in the RIT and CIT which correspond to the relation created by the process issuing the message.

a. Process ENDLOCAL messages

- Update the RIT entry corresponding to this message. Set the Length field with the value returned in the message, and set the Status field to FREE.
- Scan the RIT. If there are any entries with the Status field equal to Busy, then go to Step 2. Else go to Step 3.

b. Process ENDJOIN, ENDUNION and/or ENDNOTINSET messages

- Update the RIT entry corresponding to this message. Set the Length field with the value returned in the message, and set the Status field to FREE.
- Remove the corresponding CIT entry from the CIT.
- Scan the CIT. If there are any entries with an i-p value of T or P, then to to Step 2. Else go to Step 3.

c. Process ENDFILESEND messages

- Locate the RIT and CIT entries associated with the logical channel ID of the message. Go to Step 4.3.

Step 3. Calculate Costs

This step removes duplicate entries in the CIT, and calculates the cost for each remaining entry. If the CIT is empty, then Function DRS2 is completed.

- a. Remove all entries in the CIT which have the same source and dest relation as a previous entry.
- b. If the CIT is empty, the RP operations requested for the NDML request have been completed, with the possible exception of an Aggregator step for the termination of a recovery unit, and the CIT is empty, the RPs that have been operating on behalf of the AP must now be stopped. Each of the RPs has, by this time, executed a local recovery unit termination as a result of the NDML request itself. The DRS must now request that the NTM terminate the RPs indicated by the entries in the RPIT. Therefore, for each entry in the RPIT, the DRS will send a termination request to the NTM, indicating the host-id and the local process id that is to be terminated.
- c. Calculate the cost for each remaining entry by multiplying the length of the source relation by the width of the source relation by the transmission cost factor. The lengths and widths are obtained from the RIT, and the transmission cost factor is obtained from the TCT. Put the cost value in the corresponding entry in the CIT.
- d. Calculate the average non-null cost in the CIT. Call this average T.

Step 4. Process Join, Union, and Not-In-Set Edges

This step selects which join, union, or not-in-set is to be performed next, updates the PITs appropriately, sends FILE-TRANSFER messages, and invokes Aggregators to perform the selected join, union, or not-in-set.

- a. Select the next join, union, or not-in-set to process.
 - Select the next lowest cost entry in the

CIT; call that entry c-i.

- If c-i , T then go to step 2. (If c-i , T, then all the actions of this stage which can run in parallel have been initiated. We must now wait for results to come in.)
- If the edge-type is JOIN or NOT-IN-SET (found in JQG) AND either the source or dest relation appears elsewhere in an CIT entry which has an edge type of UNION, then go to Step 4.1.
- If the status of either the source of dest relation (found in RIT) is not FREE, then go to Step 4.a.

b. Update the PITs

- Change the status field of the RIT, for the source and dest relation entries, to BUSY.
- Remove the other entry in the CIT with the same edge-id.
- Add a new entry to the RIT, which corresponds to the results of the join, union, or not-in-set to be performed. Create a unique file name for the results, and enter it into the entry. Set the status to BUSY.
- Add an RFT entry for each attribute which will appear in the result relation. All fields of both the source and dest operands, except join or not-in-set fields, will appear in the result relation. For each join or not-in-set field, scan the APL for an entry which contains it. If it appears in the APL entry other than the current one, then it must appear in the result relation. Calculate the width of the resultant relation, and place the value in its RIT entry.
- Change all Rel fields in the JQG, CIT, and APL whose value equals either the source or

dest relation RIT index, to the RIT index of the result relation.

c. Send FILE-TRANSFER, JOIN, UNION and/or NOT-IN-SET messages

- If the source relation is not at the same site as the dest relation, send the source file. Update the appropriate RIT and CIT entries with Logical Channel ID associated with the file transfer, and change the Host ID field appropriately. Set the i-p field in the current CIT entry to T. Go to Step 4.a.
- Initiate the appropriate Aggregator to perform the join, union, or not-in-set operation, as designated by the edge-type. If the Aggregator is to run on the same node as the user AP and if an LSC Aggregator is not already running, initiate the Aggregator via the LSC; otherwise, initiate it via the NTM. Update the appropriate RIT and CIT entries with the logical channel ID associated with the join. Set the i-p field of the current CIT entry to P. Set the Host-id field in the RIT appropriately. The formats of the JOIN and UNION messages are described in Section 3.2.2.3; that of the NOT-IN-SET message, in AGG1.
- If we got to Step 4.c as a direct result of an ENDFILESEND message, then go to STEP 2, else go Step 4.

3.2.2.3 Outputs

The outputs of this function are the FILE-TRANSFER, JOIN, UNION, and NOT-IN-SET messages, and NTM requests for process termination.

a. A FILE-TRANSFER message has the following format:

FILE-TRANSFER stage-id from-site to-site rel1
rel2

where:

from-site - the current site location of
the relation
to-site - the destination site for the
relation
rel1 - the file name of the relation to
be sent
rel2 - the file name of the relation on
the to-site

b. A JOIN message has the following format:

JOIN rel1 rel2 result APL rel1-rft rel2-rft
result-rft

where:

rel1 - the file name of one of the
relations to be joined
rel2 - the file name of the other
relation to be joined
result - the file name of the resultant
relation
APL - An attribute pair list of the
join field pairs for this
join.
rel1-rft - An RFT for the fields of Rel1
rel2-rft - An RFT for the fields of Rel2
result-rft - An RFT for the fields of the
result relation

c. A UNION message stage has the following format:

UNION rel1 rel2 result rft

where:

rel1 - the file name of one of the
relations to be unioned
rel2 - the name of the other relation to
be unioned
result - the file name of the resultant
relation

rft = An RFT for the fields of the
 rel1, rel2, and result relations.

- d. See AGG1 for the format of the NOT-IN-SET message.
- e. An NTM message requesting process termination has the following format:

STOP host-id RP-process-id

where:

host-id = id of the host upon which
 the RP is running

RP-process-id = local process id of the RP
 that is to be stopped

3.2.2.4 Internal Data Requirements

Working data requirements include the following performance information tables (PITs):

- a. Transmission Cost Table (TCT):

The TCT contains transmission rates between each pair of application clusters and has the following format:

ac-1 ac-2 cost

This table should already exist in a file at each site.

- b. Relation Information Table (RIT):

The RIT contains information about each relation in the transaction, with the format:

rel-id length width status log-ch Host-ID

where:

rel-id = The file name where the relation
 resides
length = number of tuples in the relation

width = size of one tuple, in bytes
status = the present status of the
 relation (FREE or BUSY)

c. Request Processor Information Table (RPIT):

The RPIT contains information about each RP that is currently operating on behalf of the AP. It has the format:

AP-id AP-process-id RP-id RP-call-type
RP-host-id RP-process-id

where:

AP-id = program-id of the AP
AP-process-id = process id of the specific
 instance of the AP on this
 host
RP-id = program id of the RP
RP-call-type = code indicating whether the
 RP is activated via the NTM
 (type = "R" for remote) or
 is called as a subroutine
 (type = "L" for local)
RP-host-id = id of the host upon which
 the RP is running
RP-process-id = process id of the specific
 instance of the RP on
 the specific host

d. Cost Information Table (CIT):

The CIT contains information regarding all possible inter-database joins and unions, with the format.

i-p edge-id source dest cost log-ch

where:

i-p = a flag to mark the join as P
 (in progress) or T (file
 heirs transferred)
edge-id = the index of the particular
 edge into the JQG

source	=	the RIT index of the relation that will be transmitted
dest	=	the RIT index of the other relation
cost	=	the cost of transmitting the source relation
log-ch	=	the logical channel ID associated with the Aggregator process which is performing the join, union, or not-in-set.

The cost value is determined by multiplying the length of the source relation from the RIT times the width of the source relation from the RIT, times the appropriate cost formula from the TCT.

e. Result Field Table (RFT):

The RFT is originally input from the Decompose CS NDML function (PRE6), but must be updated to keep track of result attributes as joins are processed.

f. CDM1 Requirements:

The CDM1 entity classes that must be accessed in order to determine the proper staging sequence to service a transaction are the following:

66	record type
67	data field

network communications rates (initial start-up cost plus cost per byte), stored in a transmission cost table (TCT).

3.2.3 Function DRS3: Initiate CS/ES Transform Processing

This function activates the CS/ES Transformer module, which was built by the Generate CS/ES Transformer (PRE8) function of the Precompiler CI, to prepare the result relation for presentation to the original requesting application process. It also notifies the application process that the query processing is completed. It is executed only for access

requests.

3.2.3.1 Inputs

The inputs to this function are:

- Indicator that the Cost Information Table (CIT) is empty
- Relation Information Table (RIT)
- ENDCONV message

The first two inputs are produced by function DRS2: Schedule Stages. The last input is produced by a CS/ES Transformer that was activated by this function. It indicates that the requested CS/ES Transform has completed.

3.2.3.2 Processing

- If the final result relation is not stored at the site of the CS/ES Transformer, a FILE-TRANSFER message must be sent to move the file. The DRS waits for the reply and updates the RIT.
- Call the CS/ES Transformer as a subroutine with an activation message as a parameter:

CONV rel1-name rel2-name length

where:

rel1-name	=	the file name which contains the results in CS format
rel2-name	=	the file name which contains the results in ES format
length	=	the number of tuples in the relation

3.2.3.3 Outputs

The outputs of this function are the following:

- The file send message, if required, to move the final result
- The activation message to the CS/ES Transformer.
- A transaction complete message, which is returned as

DS 620141310
1 November 1985

a parameter to the application process requesting the transaction. This message includes the file name of the results.

3.3 Special Requirements

Principles of structured design and programming will be adhered to.

3.4 Human Performance

Not applicable.

3.5 Database Requirements

Not applicable.

3.6 Adaptation Requirements

The system will be implemented at the ICAM IISS Test Bed site located at the General Electric facility in Schenectady, NY. The first Distributed Request Supervisor process will be implemented on the VAX VMS host.

SECTION 4

QUALITY ASSURANCE PROVISION

Among the tests that should be incorporated into the software are:

- a. input data checks
- b. interface data checks, i.e., tests to determine validity of data passed from calling routine
- c. database verification
- d. operator command checks
- e. output data checks

Not all tests are required in all routines, but error checking is an essential part of all software.

The CI quality assurance provisions must consist of three levels of test, validation and qualification of the constructed application software.

- a. The initial level can consist of the normal testing techniques that are accomplished during the construction process. They consist of design and code walk-throughs, unit testing, and integration testing. These tests will be performed by the design team which will be organized in a manner similar to that discussed by Weinberg in his text on software development team organization (THE PSYCHOLOGY OF COMPUTER PROGRAMMING, Van Nostran Reinhold, 1971). Essentially a team is assigned to work on a sub-system or CI. This approach has been referred to as "adaptive teams" and "egoless teams." Members of the team are involved in the overall design of the sub-system; there is better control and members are exposed to each other's design. The specific advantage from a quality assurance point is the formalized critique of design walk-throughs which are a preventive measure for design errors and program "bugs." Structured design, design walk-throughs and the incorporation of "antibugging" facilitate this level of testing by exposing and addressing problem

areas before they become coded "bugs."

- b. Preliminary qualification tests of the CI are performed to highlight the special functions of the CI from an integrated point of view. Certain functional requirements may require the cooperative execution of one or more modules to achieve an intermediate or special function of the CI. Specific test plans will be provided for the validation of this type of functional requirement including preparation of appropriate test data. (Selected functions from 3.2 must be listed).
- c. Formal Qualification Test will verify the functional performance of all the modules, within the CI as an integrated unit, that accept the specified input, perform the specified processes and deliver the specified outputs. Special consideration must be given to test data to verify that proper interfaces between modules have been constructed.

DS 620141310
1 November 1985

SECTION 5

PREPARATION FOR DELIVERY

The implementation site for the constructed software will be the ICAM Integrated Support System (IISS) Test Bed site located at General Electric in Schenectady, NY. The required computer equipment will have been installed. The constructed software will be transferred to the IISS system via appropriate storage media.

END

7-87

DTIC